8-20-1992

Practical constraints pertinent to the design of neural networks

Said Sadek Abdallah

Rufus H. Cofer

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Practical constraints pertinent to the design of neural networks

Said Abdallah
Rufus H. Cofer

SPIE.

Practical Constraints Pertinent To The Design Of Neural Networks

Said Abdallah and R. H. Cofer

Florida Institute of Technology , Department of Electrical Engineering
Melbourne , Florida 32905

ABSTRACT

In designing a feedforward neural network for numerical computation using the backpropagation algorithm it is essential to know that the resulting network has a practical global minimum, meaning that convergence to a stationary solution can be achieved in reasonable time and using a network of reasonable size. This is in contrast to theoretical results indicating that any square-integrable(L2) function can be computed assuming that an unlimited number of neurons are available. A class of problems is discussed that does not fit into this category. Although these problems are conceptually simple, it is shown that in practice convergence to a stationary solution can only be approximate and very costly . Computer simulation results are shown, and concepts are presented that can improve the performance by a careful redesign of the problem .

INTRODUCTION

The output of a perceptron is the result of applying an activation function to the output of a linear combiner. Its connection weights and bias can be adapted using a variety of learning algorithms. Rosenblatt [1] developed a procedure for updating the weights when the activation function is a quantizer and proved its convergence if the inputs presented to the perceptron belong to two linearly separable classes. Widrow and Hoff [2] developed the least-mean-square algorithm (LMS) which can be applied to a linear combiner without quantization, but this linear rule may fail to separate patterns that are linearly separable [3]. Both of these algorithms fall under the category of error correction rules in the sense that they minimize the difference between the desired response and the output. With the development of the backpropagation algorithm [4] for the training of feedforward neural networks with one or more layers of neurons (perceptron nodes) between input and output layers it has become possible to solve decision problems with complex boundaries whereas single perceptrons can only separate regions by a hyperplane. One of the strengths of backpropagation networks is in its theoretical background such as "the backpropagation theorem" [5] which guarantees existence of a three layer network that can approximate any L2 function to any desired accuracy. However the theory is not constructively specific when it comes to a learning strategy, the size of the network or the activation function.

Rosenblatt's perceptron guarantees convergence and provides the ability to create sharp boundaries between input patterns, but is limited to solving linearly separable problems. Multilayer feedforward networks with backpropagation of the errors were introduced to overcome this difficulty. This meant changing the learning algorithm to a steepest-descent rule which minimizes the mean-square error over the training set and differentiable monotonically increasing activation functions became necessary, the most popular being the hyperbolic tangent. These choices have meant a significant change in the behavior of the network when solving classification problems.

This paper is an empirical study of some of the difficulties introduced by these choices, specifically losing the ability to easily implement sharp decision boundaries as well as problems introduced by the saturation region of squashing activation functions .

0-8194-0871-9/92/$4.00

THE LINEAR CLASSIFIER

Simple one layer perceptrons are very useful in mapping similar input patterns to similar output patterns and they can achieve sharp decision boundaries between regions of the input space. On the other hand networks with hidden units are needed when the decision boundaries are more complex. As a first step the classification capabilities of the one layer perceptron with a hard limiter as an activation function and the perceptron convergence procedure (an error correcting rule) for adapting the weights is compared to the one layer perceptron with a sigmoidal activation function and a steepest-descent rule for weight adaptation.

The basic building block in most feedforward neural networks is the one layer perceptron also referred to as the "adaptive linear element " or Adaline [2], shown in figure 1 for two inputs. This simple net has the ability to learn whether an input belongs to one of two classes as long as they are linearly separable. The single node computes a weighted sum of the two inputs adds a bias and passes the result through a hard limiter. The output of the linear combiner is described by :

$$Y = X1\,W1 + X2\,W2 + b \qquad (1)$$

The decision boundary is the line Y = 0 which separates the input space into two regions with the output being +1 on one side and -1 on the other. An example of the mapping of the input space to the output space is shown in figure 2 where W1 = 1 , W2 = 0 , b = - 0.6 and the two inputs vary between -1 and +1. As long as the input samples belong to linearly separable regions a sharp decision boundary can be achieved.

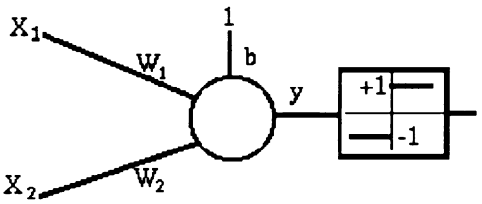

Figure 1. *Adaline with hard limiter*

If we replace the hard limiter in figure 1 by a sigmoidal function of the form :

$$f(x) = \frac{(1 - e^{-sx})}{(1 + e^{-sx})} \qquad (2)$$

and adapt the weights using the backpropagation algorithm as described in [4] (without momentum), the resulting mapping of input space to output is shown in figure 3. The goal in the above problem is to have an output of 0.5 when X1 > 0.6 and an output of -0.5 when X1 < 0.6, the slope of the sigmoid is 1 and the learning rate is set to 0.1. It is obvious that one node with a sigmoidal nonlinearity can only approach the sharp decision boundary of figure 2 as a limiting case as the slope approaches infinity but this is not practical because it can slow down the learning considerably and in extreme cases the learning can become unstable because the derivative of the sigmoid approaches an impulse function.

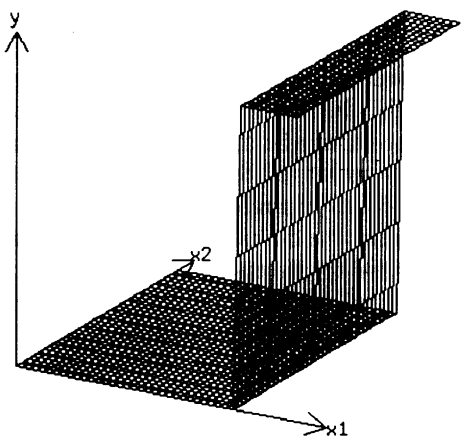

Figure 2. *Input space versus output*

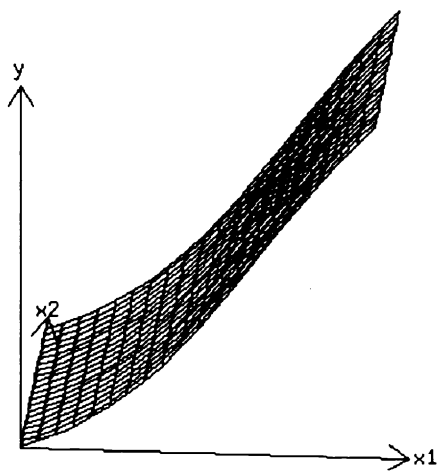

Figure 3. *Input space versus output*

Figure 4 is a plot of the mean-square error (MSE) versus number of presented input samples. The plot is generated by calculating the average over 10000 input samples (raster scan of the input space) of the square of the error every 500 iterations. Irrespective of the number of input samples presented, different initial weights and different learning rates, convergence towards a specific solution is never achieved. As seen from figure 4 the value of the MSE and the line that separates the input space into two regions continue to oscillate.

For the perceptron with a hard limiter the problems of minimizing the MSE and minimizing the number of wrong classifications have a common solution. This is not the case for the sigmoidal perceptron with backpropagation learning. For a general problem the minimum that the steepest-descent rule is seeking does not necessarily minimize the number of classification errors. This is exactly the reason for the observed oscillations in figure 4 because steepest-descent leaves a certain number of input samples that are being missclassified which prevents the convergence to the minimum. This occurs when the weights are being updated after every input sample is presented. The oscillations can be considerably reduced by applying what is known as batch updating (Accumulating the weight correction terms for several patterns before making a single weight adjustment). The important fact remains that a clear distinction exists between minimizing MSE over the entire training set and minimizing missclassifications. As an example using batch updating (every 200 input samples) the weights and bias converge approximately to W1 = 1.83 , W2 = 0.05 and b = - 0.7. This results in a boundary that is quite different from the desired line, X1 = 0.6. This holds true independent of the initial weights and learning parameters.

Figure 4. *MSE versus number of iterations*

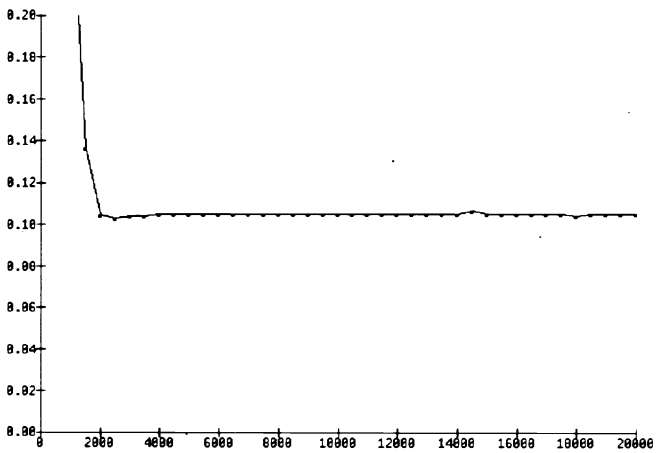

Figure 5. *MSE versus number of iterations*

If our objective is to adapt the weights to a solution that minimizes the number of classification errors we can change the learning strategy slightly by allowing the updating of the weights only when the sign of the desired input and that of the actual output are different, in other words update only if an input pattern is missclassified. This changes the goal of the training algorithm from finding the minimum of the error surface to correcting for errors in classification, which transforms our problem into a much easier one where the task is to have the output of the net equal to zero when the inputs belong to the boundary between the two regions. This problem can be solved by any combination of weights and bias that result in the line, X1 = 0.6. The results of this strategy are shown in figure 5 which is a plot of MSE versus number of input samples. Comparing this to figure 4 the oscillations are much reduced and the convergence is to a solution with a higher MSE. The weights converge to different values for different initial conditions, but always results in the desired boundary and always to a MSE higher than the Minimum of the Error surface. Table one shows the results of several such runs.These general results are observed for other similar problems (except when the desired boundary passes through the origin of the input space). In order to achieve the response shown in figure 2, a hard limiter can be applied to the output after training.

Table 1.

W1	W2	b	line
4.96	0	-2.98	X1=0.6
0.537	0	-0.32	X1=0.59
2.97	0	-1.76	X1=0.59
0.5	0	-0.9	X1=0.6
5.16	0	-3.1	X1=0.6

These results are achieved without having to change the basic algorithm and at a reduction in the amount of computation since errors are backpropagated and weights updated only in the event of continually decreasing missclassifications.

NETWORKS WITH HIDDEN LAYERS

Although hidden layers with their internal representations allow for decision boundaries that go far beyond simple hyperplanes, at the same time the ease with which a single perceptron with a hard limiter can create sharp boundaries in the input space is lost. This fact creates the distinction, as illustrated in the previous sections for the adaline, between minimizing MSE and minimizing the number of classification errors. If a network with hidden layers can implement sharp boundaries the difference between these two problems is removed. What follows illustrates that this is not the case. Sharp transitions in the output due to very small changes in the input present a very difficult task that can only be approached at great expense.

Let figure 6 be a desired mapping from input space (two inputs X1 and X2 varying between -1 and +1) to the output space (one output y). The desired output is +0.5 whenever the input sample is within a circle of radius 0.5 , and -0.5 otherwise. This describes what is known as the "circle in a square" problem. First a network with one hidden layer (twelve nodes) is trained using the backpropagation algorithm (zero momentum, learning rate of 0.1 and batch updating every 20 input samples).Figures 7 and 8 are plots of MSE versus number of input samples and the mapping of the input space to the output respectively. This is still quite far from the desired response of figure 6, and as figure 7 shows the MSE levels off at a certain value and extra training does not cause it to drop any further. At this stage (with 100000 input samples presented) the probability of successful classification is 91%. The other option available for improving the performance is increasing the size of the network. Figures 9 and 10 show the results for a network with two hidden layers each with 60 nodes, the probability of successful classification reaches 97%, but only after 150000 input samples. Adding even more layers and nodes leads to only slight improvements. The issue here is not whether we can improve on these results by choosing different training parameters or by different versions of the learning algorithm, it is the fact that a relatively simple problem presents practical difficulties for this type of network. These difficulties are mostly due to the requirement that a very small change in input should cause a large jump in output. This becoming very evident in the "circle in a square" problem if we introduce a don't care region between the two classes. It is found that as the width of this region is increased the difficulty of the problem decreases at a very fast rate.

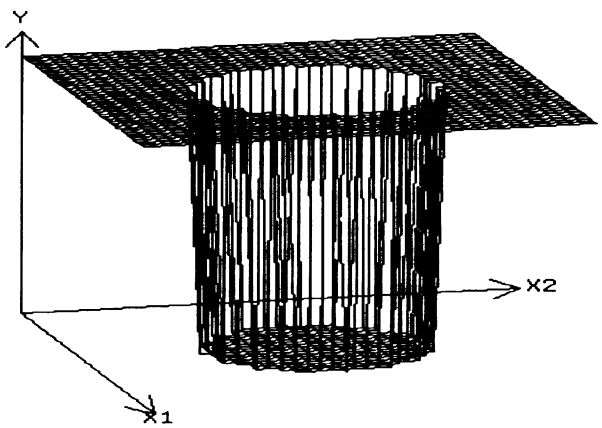

Figure 6. *Input space versus output*

Going back to the problem of classification the approach introduced in the previous section is used to solve the current problem where the network is trained only when an error in classification is made. The same parameters for the training algorithm are used but without batch updating, the results are shown in figures 11 and 12

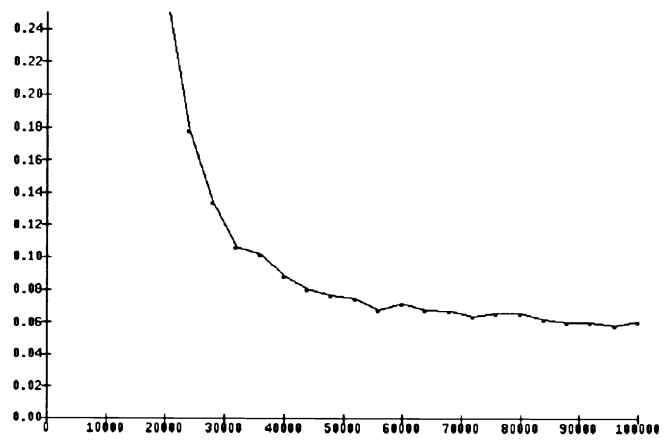

Figure 7. *MSE versus number of iterations*

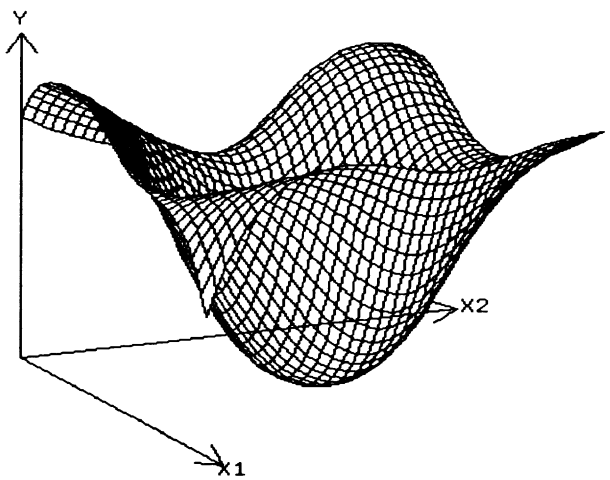

Figure 8. *Input space versus output*

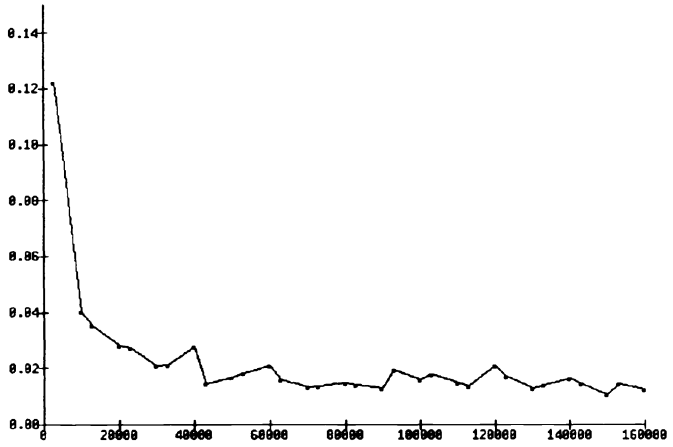

Figure 9. *MSE versus number of iterations*

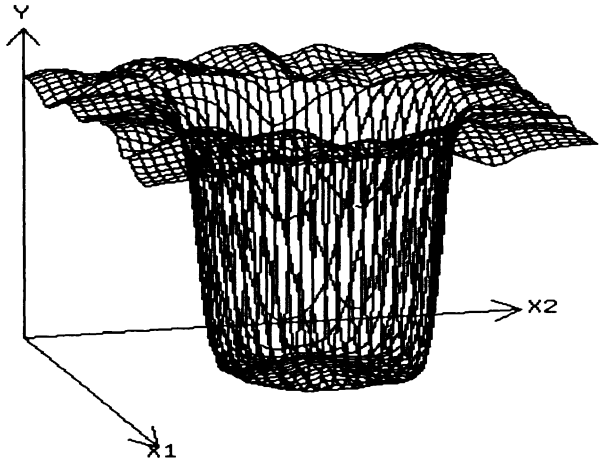

Figure 10. *Input space versus output*

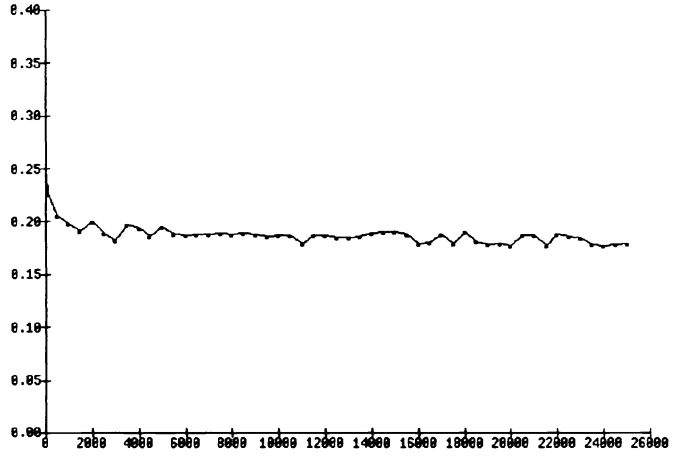

Figure 11. *MSE versus number of iterations*

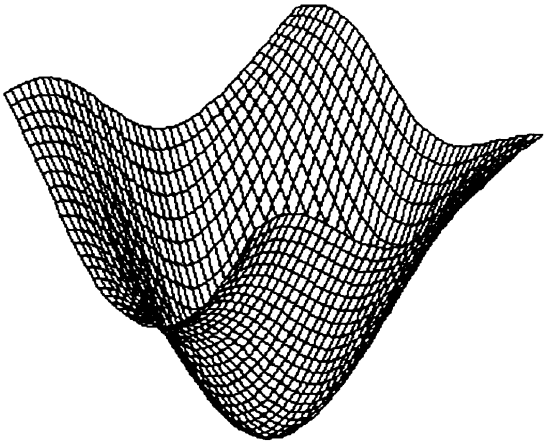

Figure 12. *Input space versus output*

for a network with one hidden layer (10 nodes). After only 1500 input samples are presented the probability of success in classification reaches 98% even though the MSE is still relatively high, continuing with the training can further improve the results to 99.5% success rate. This is achieved even though the mapping of the input space to the output, as shown in figure 12, is nowhere near that of figure 6. Applying a hard limiter to the output after training results in the decision boundary shown in figure 13.

Although an approach is found that works well for the classification problem, thus indirectly leading to a sharp decision boundary by applying a hard limiter to the output after training. It remains an interesting result that sharp transitions in the output in response to relatively small changes in the input can not be directly and practically achieved by the backpropagation network. This is a good result if we are looking for a network that can generalize but not so good if our objective is to perform such separations.

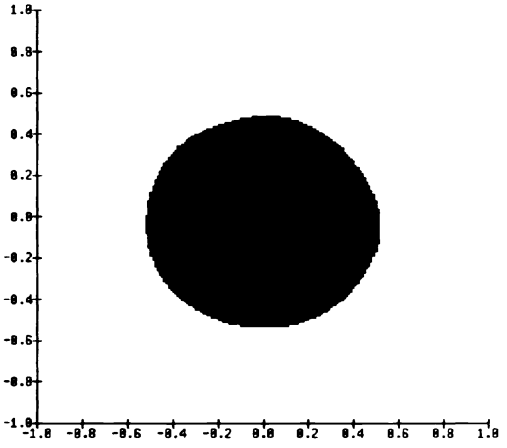

Figure 13. *Decision boundary achieved by the neural network*

SQUASHING ACTIVATION FUNCTIONS

The backpropagation algorithm requires activation functions that are differentiable and monotonically increasing. The popular choices are sigmoidal functions that are also known as squashing functions due to the effect of their saturation regions. Equation (2) is an example of such a function. Since backpropagation is a steepest descent method that seeks the minimum of the error surface the general shape of these surfaces plays an important role in the degree of success or failure of the algorithm. These error surfaces in turn derive a lot of their characteristics from the chosen activation function.

To demonstrate the nature of MSE surfaces a simple experiment with the adaline of figure 1 was performed. The adaline was driven by a typical set of inputs and desired outputs and the MSE was plotted as a function of the two weights. Figure 14 shows the error surface when the activation function is the identity and figure 15 shows the error surface when the activation function is the sigmoid in equation (2) , the height of the surface at each point is the MSE. Although looking at these surfaces it is not possible to arrive at a proof of the superiority of one function over the other it is still possible to infer that one surface might be more suited to steepest descent methods. The paraboloid in figure 14 is the obvious choice but for the limitation that it is only applicable to linearly separable problems. This leads us to the need for nonlinear activation functions.

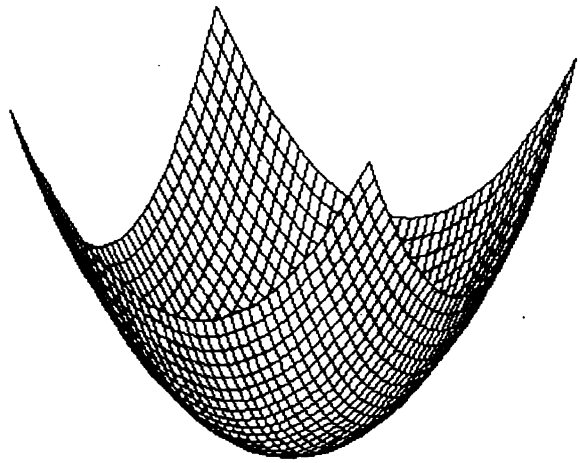

Figure 14. *Typical MSE surface of a linear combiner*

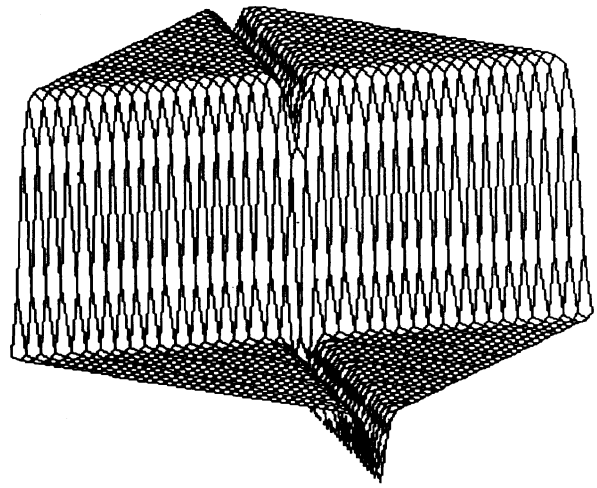

Figure 15. *Typical sigmoidal MSE surface*

It is widely known that squashing activation functions introduce two specific difficulties. Local minima and what are known as flat regions of the error surface. These regions are the topic of discussion in the following paragraphs.

The use of squashing activation functions leads to error surfaces with extensive flat areas and little slope along one or more weight dimension. These areas are a result of the input vector and the weights of a node being such that the output of the linear combiner falls in the saturation region of the activation function where the derivative is very small. Since the weight update term is proportional to this derivative the weights will update very slowly even for relatively large errors. Unfortunately this can not be overcome by simply increasing the learning rate since the error surface also has regions that are fairly steep and adjusting the weights by a large amount may lead to overshooting of the minimum.

This problem has been addressed several times [3,4,5,8] and a variety of solutions exist ranging in difficulty and effectiveness, but all these solutions have one point in common. They assume that squashing activation functions are the only available option and they tend to work around the problems they cause.

The mathematics of backpropagation networks does not demand that the output of the activation function saturates beyond a certain point. Consequently it is not very clear why such functions are almost exclusively used given the irregularity of the error surfaces they produce. If the purpose is to approach the classification capabilities of the hard limiter it is obvious from the pervious section that the saturation region does not help towards that end. In practice it is recommended [4] that desired outputs be picked from outside the saturation region specifically to avoid the flat areas.

As networks grow in size so does the likelihood of any one unit getting stuck in such a region, this effectively limits the output of these units to the two saturation levels of the nonlinearity during long periods of training. Specially if we consider units in the hidden layers there is no advantage in limiting there outputs in this way. Equation (3) is an example of a non-saturating activation function and figure 16 is a typical error surface (generated using same input output patterns used to generate figure 15).

$$f(x) = \begin{cases} \log(1+x) & \text{if } x>0 \\ -\log(1-x) & \text{if } x<0 \end{cases} \qquad (3)$$

The derivative is given by :

$$f'(x) = \begin{cases} 1/(1+x) & \text{if } x>0 \\ 1/(1-x) & \text{if } x<0 \end{cases} \qquad (4)$$

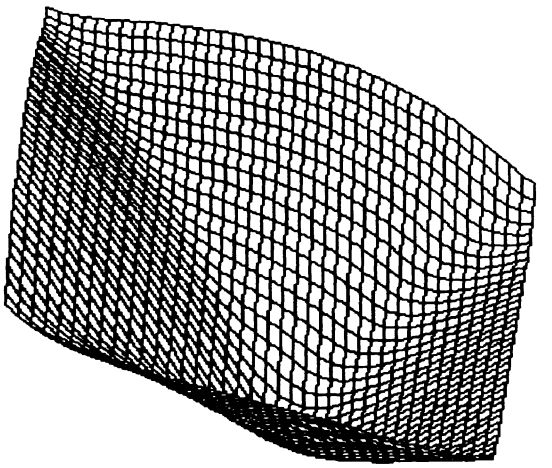

Figure 16. *Typical logarithmic MSE surface*

Note that the derivative is continuous at x = 0, its value goes to zero as x increases in magnitude but at a much slower rate than the derivative of the hyperbolic tangent and it is just as easy to compute. Figure 17 is a graph of the log function of equation (3) while figure 18 is a graph of its derivative equation (4) (for x > 0).

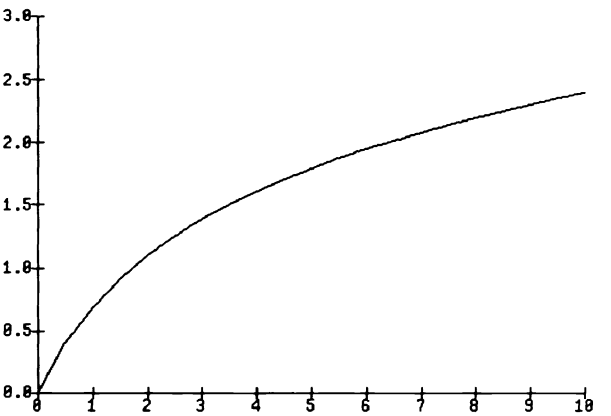

Figure 17. *Log activation function (x>0)*

In a network with more than two weights it is possible to look at slices of the error surface by varying two weights and plotting the MSE while holding all other weights constant. Typical surfaces for the sigmoid and the 'log' function are shown respectively in figures 19 and 20. The flat regions typical of sigmoidal error surfaces do not appear in the error surfaces of the new activation function.

Looking at error surfaces can be very helpful in understanding some of the difficulties that are encountered in training a backpropagation network. This understanding remains mostly intuitive until it is backed by experimental results. For this purpose two problems are solved using both activation functions (2) and (3), with the understanding that the results can not be conclusive until more exhaustive experiments are performed.

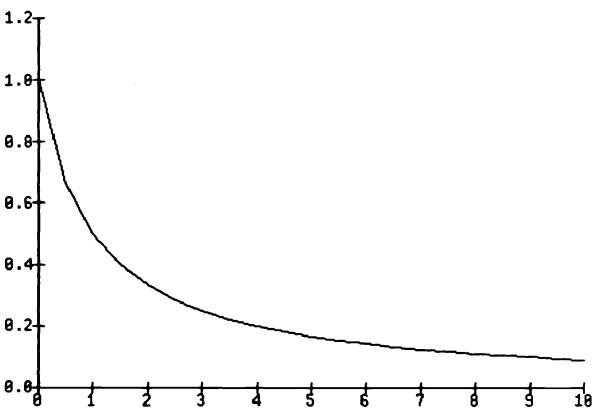

Figure 18. *Derivative of log function (x>0)*

The first task known as the parity problem can be described as follows : The network has four bipolar inputs, one hidden layer with six nodes and one output. (Six hidden units are used instead of the necessary four to reduce the effect of the local minima). The desired output is -0.8 whenever the number of 1's in the input is odd and +0.8 otherwise. the initial weights are picked randomly between -1 and +1 and for each set of initial weights the network is trained with different combinations of the activation function and learning rate. This was done for three initial weight sets which generated the results in tables 2 and 3. Training is considered complete when the MSE drops below 0.05.

The second problem falls in the category of function approximation, where the function is f(x)=sin(x).sin(y), x

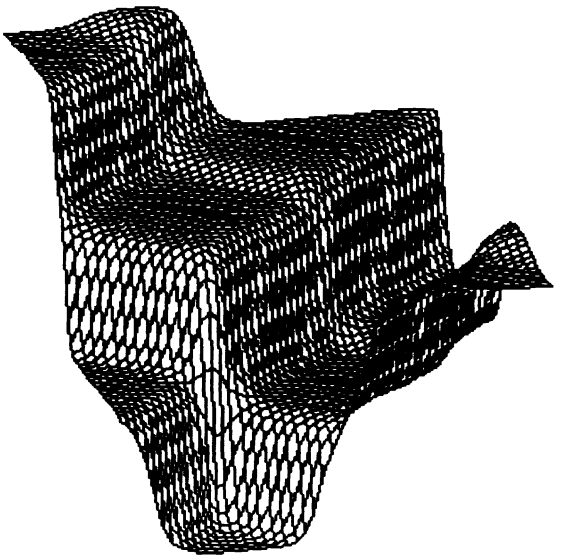

Figure 19. *Typical sigmoidal MSE surface*

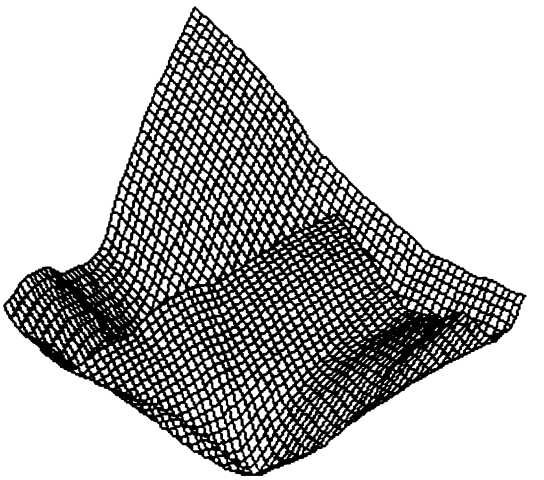

Figure 20. *Typical logarithmic MSE surface*

and y being the two inputs ranging between zero and Π. The network has 8 units in the hidden layer and one output. Training proceeds as in problem one and tables 5 and 6 contain the results. Training is considered complete when MSE drops below 0.02. Both experiments were performed without momentum and the updating of the weights was after every presentation of an input pattern.

From examining the results two main advantages of a non-saturating activation function become apparent. The first has to do with the number of iterations needed before training is complete. In problem 1 the average number of

Table 2

Learning rate	# of required iterations		
0.05	415	355	332
0.1	211	459	669
0.15	471	x	x
0.2	x	x	x
0.25	x	x	x
0.3	x	x	x

Table 3

Learning rate	# of required iterations		
0.05	1297	6200	1313
0.1	629	825	x
0.15	423	466	x
0.2	335	774	x
0.25	303	x	x
0.3	x	x	x

Table 4

Learning rate	# of required iterations		
0.05	1958	2183	3569
0.1	1156	1089	1726
0.15	2411	1208	2090
0.2	2185	1328	2445
0.25	3169	3042	3194
0.3	3692	3793	3448

Table 5

Learning rate	# of required iterations		
0.2	3899	1318	3120
0.3	7727	967	3224
0.4	3094	6229	5053
0.5	1626	8560	3741
0.6	1522	2968	3555
0.7	1536	2558	3645
0.8	1761	1552	4190

iterations is approximately 300 for the log and 694 for the sigmoid. In problem 2 the numbers are 1323 for the log and 1870 for the sigmoid. The second advantage has to do with the sensitivity of the network to the initial weights and to the learning rate. It is evident from the results that the sigmoidal network is highly sensitive to these parameters. For example looking at table 5 it is difficult to decide which learning rate is best for that problem, since starting from different initial weights the best performance is achieved using three different learning rates and with a substantial difference in the number of iterations required in each case. On the other hand the results in tables 2 and 4 show that sensitivity to these parameters when using the log function is reduced by a considerable amount . For example the best learning rate to use in problem 2 is easily found to be 0.1 since it works best for most situations and is not dependent on the initial weights.

It is important to note that the presented results are only a small portion of all the experiments performed. The conclusions in the above paragraph are based on a larger set of problems and computer runs.

CONCLUSION

The task of separating input patterns that can be arbitrarily close to each other into different classes is shown to be a practically difficult task for the backpropagation network. A learning strategy is introduced that can avoid this difficulty whenever the goal is that of classification.

Problems introduced by squashing activation functions were discussed and an alternative to these functions was presented. The experimental results using the new function showed an improvement in learning speed and a reduction in the sensitivity of the network to the parameters of the learning algorithm.

REFERENCES

[1] F. Rosenblatt, "Principles of Neurodynamics ". New York:Spartan 1962.

[2] B. Widrow and M. Hoff , "Adaptive Switching Circuits" IRE WESCON, Conven. Rec. , Sept 1960, pp 96-104, part 4.

[3] B. Widrow and M. Lehr, "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation ." , Proceedings of The IEEE, Vol 78, No. 9, September 1990, pp 1415-1442.

[4] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Propagation.", in "Parallel Distributed Processing: Explorations In The Microstructure of Cognition ." , D. Rumelhart and J. McClelland, Eds. Cambridge, MA: M.I.T. press 1986.

[5] R. Hecht-Nielsen , "Neurocomputing. " , Addison-Wesley 1990.

[6] R. Lippmann, "An Introduction to Computing With Neural Nets.", in IEEE ASSP magazine, April 1987, pp 4- 22.

[7] R. Jacobs , "Increased Rates of Convergence Through Learning Rate Adaptation ", in Neural Networks, vol 1, 1988, pp 295-30 .

[8] S. Fahlman, "An Empirical Study of Learning Speed in Back-propagation networks", Tech. Report CMU-CS-88-162, Carnegie Mellon University, June 1988.